Online Inference for Adaptive Diagnosis via Arithmetic Circuit Compilation of Bayesian Networks

Sara Zermani, Catherine Dezan, Reinhardt Euler and Jean-Philippe Diguet
Lab-STICC, CNRS UMR 6285, Université de Bretagne Occidentale
Sara.Zermani@univ-brest.fr

I. Introduction and context

Considering technology and complexity evolution the design of fully reliable embedded systems will be prohibitively complex and costly. Onboard diagnosis is a first solution that can be achieved by means of Bayesian networks [1], [2]. An efficient compilation of Bayesian inference is proposed in [3] using Arithmetic Circuits (AC). ACs can be efficiently implemented in hardware to get very fast response time. This approach has been recently experimented in Software Health Management of aircrafts or UAVs [4]. However, there are two kinds of obstacles that must be addressed. First, the tree complexity can lead to intractable solutions and second, an offline static analysis cannot capture the dynamic behaviour of a system that can have multiple configurations and applications.

In this paper, we present our direction to solve these issues. Our approach relies on an adaptive version of the diagnosis computation for different kinds of applications/missions of UAVs. In particular, we consider an incremental generation of the AC structure. This adaptive diagnosis can be implemented using dynamic reconfiguration of FPGA circuits.

II. Compilation of a Bayesian network and inference computation

Our work makes use of [3] and [5] as follows: each Bayesian network [6] can be represented as a multi-linear function (MLF). This MLF is transformed into an arithmetic circuit (AC) and the probabilities are computed using this AC. The construction of the AC is done offline but the implementation is different and all conditional probabilities can be computed online as soon as the evidences (indicator values) are given.

A. Bayesian Networks as MLFs

For each Bayesian network we can define a unique MLF over two types of variables:

- Evidence indicators (λ_x): representing the members of a set of evidences (binary values).
- Network parameters ($\theta_{x|u}$): representing the parameters concerning the probability values associated with the variables X and U.

$$f = \sum_{x} \prod_{xu \sim x} \lambda_x \theta_{x|u} \tag{1}$$

The Bayesian network of Fig.1.a) relies on two variables A and B which have two states, a and $\overline{a}$ for A, b and $\overline{b}$ for B. Consequently, the MLF of this network is as follows:

$$f = \lambda_a \theta_a \lambda_b \theta_{b|a} + \lambda_a \theta_a \lambda_{\overline{b}} \theta_{\overline{b}|a} + \lambda_{\overline{a}} \theta_{\overline{a}} \lambda_b \theta_{b|\overline{a}} + \lambda_{\overline{a}} \theta_{\overline{a}} \lambda_{\overline{b}} \theta_{\overline{b}|\overline{a}}$$

To compute the margin probability at evidence e (*P(e)=f(e)*), each evidence indicator λ_x is replaced in f by 1 if x is consistent with e, and by 0 otherwise. For example, if the evidence e is $\overline{b}$, *f(e)* is computed by setting $\lambda_{\overline{b}}$=1, λ_b=0, λ_a=1 and $\lambda_{\overline{a}}$=1. Then

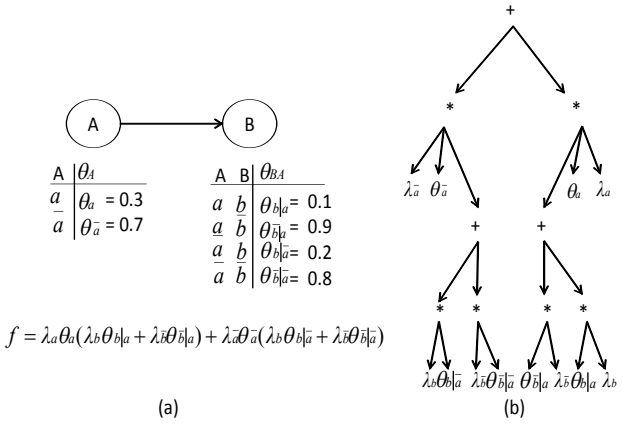

Fig. 1. a) A Bayesian network with the factored function. b) The corresponding arithmetic circuit

we get : f($\overline{b}$)= f(λ_a=1, $\lambda_{\overline{a}}$=1, λ_b=0, $\lambda_{\overline{b}}$=1)= θ_a * $\theta_{\overline{b}|a}$ +$\theta_{\overline{a}}$ * $\theta_{\overline{b}|\overline{a}}$= 0.3* 0.9+ 0.7* 0.8= 0.83.

From Bayes' theorem, we have *P(x|e)*= $\frac{P(x,e)}{P(e)}$. We can see that *P(x,e)*= f(e,λ_x=1, $\lambda_{\overline{b}}$= 0) or *P(x,e)*= $\frac{\partial f}{\partial \lambda_x}(e)$ (x not included in e).

Computing probabilities by means of MLFs explodes when the number of variables becomes large. For this reason, it is proposed to compile the network into an AC in order to reduce time and space.

B. Compilation into an AC

An arithmetic circuit is a compact graphical representation of a function f over variables (see Fig.1.b)).

There are different ways to compile an MLF into an AC [7], [8]. We have chosen the classical one based on factorisation and the jointree method. We present a hierarchical method of the AC construction.

- Arithmetic circuit by factorisation [5]
  In the example of Fig.1.a): f=λ_a θ_a λ_b $\theta_{b|a}$ + λ_a θ_a $\lambda_{\overline{b}}$ $\theta_{\overline{b}|a}$ + $\lambda_{\overline{a}}$ $\theta_{\overline{a}}$ λ_b $\theta_{b|\overline{a}}$+ $\lambda_{\overline{a}}$ $\theta_{\overline{a}}$ $\lambda_{\overline{b}}$ $\theta_{\overline{b}|\overline{a}}$, and by factoring over *A* we obtain: f= λ_a θ_a (λ_b $\theta_{b|a}$+ $\lambda_{\overline{b}}$ $\theta_{\overline{b}|a}$) + $\lambda_{\overline{a}}$ $\theta_{\overline{a}}$ (λ_b $\theta_{b|\overline{a}}$+ $\lambda_{\overline{b}}$ $\theta_{\overline{b}|\overline{a}}$).
  To have the AC of the factorised fonction, the leaves of the arithmetic circuits are λ, θ, and the nodes of the tree represent a multiplication (addition), respectively. The Bayesian network and its AC structure is given in Fig.1. The factorization is simple to obtain but it takes an exponential space.
- Arithmetic circuit by jointree [3]
  The goal is to generate the smallest possible circuit from a given jointree. After the choice of a root cluster, the

arithmetic circuit is generated from the jointree (set of clusters and separators) as follows:
  - one output addition node f;
  - one addition node for each instantiation of a separator;
  - one multiplication node for each instantiation of a cluster;
  - one input node (λ_x) and ($\theta_{x|u}$) (for all parents of x).
- Hierarchical building of the AC structure We observe that an Incremental building of AC through jointree structure can be realised. A simple illustration is given in Fig. 2.

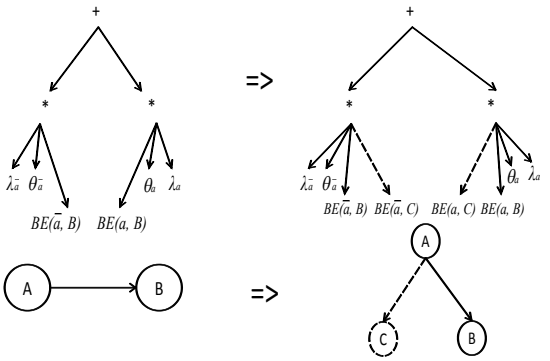

Fig. 2. Hierarchical building of the AC

We propose to take advantage of this technique that can be applied to kinds of Bayesian networks (or jointrees). This hierarchical building avoids to store all AC structures corresponding to the possible Bayesian networks built for the diagnoses of the different applications running on the embedded system. Considering the exponential growth of the system this point is crucial.

C. Computing probabilities using AC

To compute *P(x|e)*, we first need to evaluate *f(e)* and then to compute the circuit derivatives to get *f(x,e)*= $\frac{\partial f}{\partial \lambda_x}$.

- Evaluating an arithmetic circuit: in an upward-pass, the value of a node in the AC tree is computed after the computation of the values of its children.
- Computing the circuit derivatives: in a downward-pass, the derivative of a node is computed after the computation of the derivative of its parent.
  Let v be an arbitrary node in a circuit f, and assume that we are interested in the partial derivative of f with respect to node v, $\frac{\partial f}{\partial \lambda_v}$. If v is the root node (circuit output), then $\frac{\partial f}{\partial \lambda_v}$=1. If v is not the root node and has parents p, then by the chain rule of differential calculus:

$$\frac{\partial f}{\partial v} = \sum_p \frac{\partial f}{\partial p}\frac{\partial p}{\partial v}$$

Suppose now that v' is another child of a parent p. If p is a multiplication node, then Equation (2) applies and if p is an addition node, then Equation (3).

$$\frac{\partial p}{\partial v} = \frac{\partial v(\prod_{v'} v')}{\partial v} = \prod_{v'} v' \quad (2) \qquad \frac{\partial p}{\partial v} = \frac{\partial v + (\sum_{v'} v')}{\partial v} = 1 \quad (3)$$

With these equations, we can recursively compute the partial derivatives of f with respect to any node v.

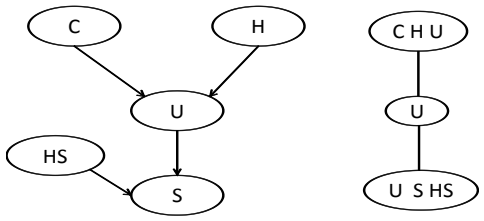

Fig. 3. The Bayesian network of a camera application (task 1) and its jointree.

Complexity: the upward-pass clearly is achieved in a time linear in the size of the circuit (number of edges in the circuit). The downward-pass takes linear time only when each multiplication node has a bounded number of children. The disadvantage of this approach is that all the computations done in the upward-pass must be saved in order to be used in the downward-pass.

In the next section, we propose a simpler method for the AC computation without any downward-pass for the diagnosis of an application. We also propose to use a reconfiguration of AC when the application and its diagnosis have to change in time.

III. The adaptive diagnosis of a UAV embedded system through missions

We want to define an adaptive diagnosis for a set of different applications (or tasks) by modifying the AC associated with the diagnosis of one of them.

A. The Bayesian network corresponding to a task

Consider a UAV including an embedded system with a task that consists in taking photos upon receipt of a demand. The system can take a photo only if is has enough memory. The scenario is the following: if it can't take a photo and it has got enough memory, or if it can take a photo and it doesn't have enough memory, then the functionality is turned to bad. We assume that the space in the memory is computed with a logical sensor, which can produce state ok or bad.

A Bayesian network associated with this task can be defined as follows:

- Nodes:
  Command(C): representing the demand of taking a photo. It takes the values Yes or No (notation (c or $\overline{c}$)).
  State(U): indicating whether the photo is taken or not. It can have the values Yes or No (notation (u or $\overline{u}$)).
  Health(H): representing the health of the system associated to the internal state U. It takes the value Ok or Bad. (notation (h or $\overline{h}$)).
  Sensor(S): indicating whether the capacity of the memory is sufficient or not. It takes the value Yes or No. (notation (s or $\overline{s}$)).
  Health-Sensor:(HS) representing the health of the Sensor. It takes the value Ok or Bad (notation (hs or $\overline{hs}$)), depending on the reliability of the sensor S.
- Edges:
  $C, H \rightarrow U$: U depends on C and H. U is commanded by C and monitored by H.
  $U, HS \rightarrow S$: S depends on H and U. U is observed by S which is monitored by HS.
  The detailed Bayesian network of this example and its jointree is given in Fig. 3.

B. Computing the Health probability $P(Health|Sensor, Command)$

The goal is to diagnose the current state of the system by evaluating the node H representing its health. This evaluation is done upon the knowledge of some evidences (the value c is the evidence associated with the command C and the value s the evidence associated to the sensor S) and it corresponds to the computation of $P(h|s,c)$.

We know that:

$$P(h|s,c) = \frac{P(h,s,c)}{P(s,c)} \qquad P(h,s,c) = \frac{\partial f}{\partial(h)}(s,c)$$

$$P(s,c) = f(s,c)$$

Considering node H as the root node of the AC, we obtain at the last step of the upward-pass:

$$f(s,c) = f(\lambda_h = 1) + f(\lambda_{\overline{h}} = 1)$$

and

$$f(h,s,c) = f(\lambda_h = 1) + f(\lambda_h = 0) = f(\lambda_{\overline{h}} = 1)$$

We can see that if we take H as the root node of the AC, we don't need the downward-pass any more. We can apply this to the other node HS. In general, for a health node HX, we compute as indicated until reaching λ_{hx} and $\lambda_{\overline{hx}}$ and with λ_{hx}=1, we get $f(e, hx)$ and by adding f(e,$\lambda_{\overline{hx}}$=1) we get $f(e)$. After this step, we continue by taking into account the two computations for each node.

C. Reconfiguration of an AC for adaptive diagnosis

Now, assume that our UAV has some missions related to different type of applications (or tasks). There are two types of task: the static one (executed from the beginning to the end of the mission, as for example path-planning) and the temporary one (executed at a certain time, as for example taking photos, then writing into a file the external and the internal temperature, and finally writing into the same file the altitude measures using both barometric and laser altimeter).

The goal is to monitor the health of the system at any time. The complete arithmetic circuit for static and temporary tasks of the whole system has to be computed in a static way before the mission. Nevertheless, some parts of the diagnosis (or AC structure) could be done more on-the-fly, especially the temporary tasks.

We propose to compile the arithmetic circuit of the static tasks offline. For each new temporary task, we reconfigure the AC by deleting the nodes of the previous task and adding the node of the current one in real time. We assume that the UAV should execute task 1 'take photos' presented in Fig. 3 for a specific time slot and in a context of other static tasks. For the temporary task 2, we look at the computation of the altitude using 3 sensors (laser sensor, barometer sensor and IMU). The Bayesian networks for the diagnosis of these two tasks and their jointrees are given in Fig. 3 and Fig. 4. Since we have a large common part within the two jointrees, we can use the AC of task 1 to obtain the complete AC of task 2. For this we use the hierarchical building as illustrated in Fig. 2.

- For the cluster (U, S-IMU), these steps are as follows:
  1) go to all (*) of U;
  2) add a (+) node for each (*) node and add an arc for BE(U|HC,S-IMU), where $U|HC$ corresponds to any instantiation of U, H and C for the computation of the probability $U|HC$.
- For the cluster (U, H-Bar, S-Bar), these steps are as follows:
  1) go to all (*) of U;
  2) add a (+) node and two (*) nodes for each (*) node of U. For the first (*) node, add an arc for λ_{H-Bar}, θ_{H-Bar} and BE(U|HC, S-Bar). For the second (*) node, add an arc for $\lambda_{\overline{H-Bar}}$, $\theta_{\overline{H-Bar}}$ and BE(U|HC, S-Bar) for the corresponding instantiation.

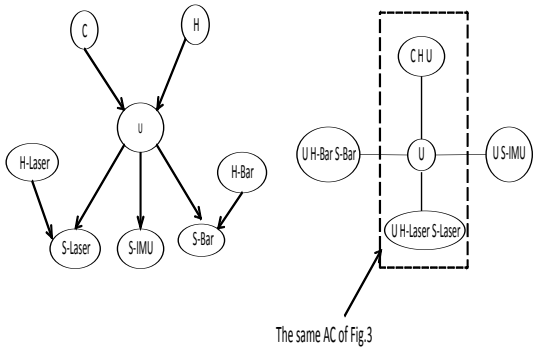

Fig. 4. The Bayesian network of an altitude application (task 2) and its jointree

This method can be generalised to any Bayesian network by adding corresponding constraints.

IV. Conclusion and perspectives

In this paper, we have applied adaptive diagnosis to a number of independent tasks. The next step will consist in applying adaptive diagnosis to tasks with interactions (e.g., being executed in the same or different contexts or sharing specific ressources). We also study the efficient hardware implementation of such configurable diagnosis engines that can offer very short response times. FPGAs provide the kind of parallelism that fit with Bayesian networks but architectures with different performances/area tradeoff are possible. We explore this design space, with two different approaches. The first one is based on partial/dynamic reconfigurations of Xilinx FPGAs. The second one is an application specific processor to be also implemented as a soft core on a FPGA.

References

[1] O. J. Mengshoel, A. Darwiche, and S. Uckun, "Sensor validation using Bayesian networks," in *Proceedings of the 9th International Symposium on Artificial Intelligence, Robotics, and Automation in Space (iSAIRAS-08)*, (Los Angeles, CA), Feb. 2008.

[2] B. W. Ricks and O. J. Mengshoel, "The diagnostic challenge competition: Probabilistic techniques for fault diagnosis in electrical power systems," in *Proceedings of the 20th International Workshop on Principles of Diagnosis (DX-09)*, (Stockholm, Sweden), 2009.

[3] A. Darwiche, "A differential approach to inference in bayesian networks," *J. ACM*, vol. 50, no. 3, pp. 280–305, 2003.

[4] J. Schumann, K. Y. Rozier, T. Reinbacher, O. J. Mengshoel, T. Mbaya, and C. Ippolito, "Towards real-time, on-board, hardware-supported sensor and software health management for unmanned aerial systems," in *Proceedings of the 2013 Annual Conference of the Prognostics and Health Management Society (PHM2013)*, October 2013.

[5] A. Darwiche, "A differential approach to inference in bayesian networks," in *UAI*, pp. 123–132, 2000.

[6] R. G. Cowell, *Probabilistic networks and expert systems: Exact computational methods for Bayesian networks*. Springer Science & Business, 2006.

[7] Z. Lian, Y. Jinsong, W. Jiuqin, and X. Wei, "Real time diagnosis with compiling bayesian networks," in *Industrial Electronics and Applications (ICIEA), 2011 6th IEEE Conference on*, pp. 542–546, June 2011.

[8] M. Chavira and A. Darwiche, "Compiling bayesian networks using variable elimination.," in *IJCAI* (M. M. Veloso, ed.), pp. 2443–2449, 2007.